Graph Representation Learning

Please use the official LaTeX template to type your answers available in Moodle. Please respect the notation from Table 1 in your answers whenever applicable:

Table 1: Notation.

σ	An element-wise non-linearity.
t	Iteration, or layer t.
$d^{(t)}$	The dimension of a vector at iteration t.
d	The dimension of a vector, and an abbreviation for $d^{(0)}$.
$\mathbf{1}^d \in \mathbb{R}^d$	A d-dimensional vector of all 1's.
$\mathbb{I}^d \subseteq \mathbb{R}^d$	The set of d-dimensional one-hot vectors.
$\mathbb{B}$	Boolean domain $\{0, 1\}$.
$\mathbf{b}^{(t)} \in \mathbb{R}^d$	A bias vector.
$\mathbf{x}_u \in \mathbb{R}^d$	The feature of a node $u \in V$.
$\mathbf{h}_u^{(t)} \in \mathbb{R}^{d^{(t)}}$	The representation of a node $u \in V$ at layer t.
$\mathbf{z}_u = \mathbf{h}_u^{(T)} \in \mathbb{R}^{d^{(T)}}$	The final representation of a node $u \in V$ after T layers/iterations.
$\mathbf{W}_x^{(t)} \in \mathbb{R}^{d^{(t+1)} \times d^{(t)}}$	Learnable parameter matrix at layer t.
MLP	A multilayer perceptron with ReLU as nonlinearity.

Question 1

(a) (For this answer, all graphs are assumed to be undirected.) We aim to construct a function f that associates to each graph G a function $f(G) : V_G \to \mathbb{B}$ such that there is no parametrization of $\mathcal{S}$ satisfying $f(G) = \tilde{\mathcal{S}}(G)$ for all graphs G.

First, we construct a graph $G = (V_G, E_G)$ that will serve us as a counter-example. Let C_3 be the cycle-graph of length 3, that is, let $V_{C_3} = \{1, 2, 3\}$ and let $E_{C_3} = \{(1, 2), (2, 3), (3, 1)\}$. Then denote $G = C_3 \dot\cup C_3$, so that G is the disjoint union of two cycle-graphs of length 3. On G, we define a function $h : V_G \to \mathbb{B}$ via

$$h : V_G \longrightarrow \mathbb{B}, \qquad h(u) = \begin{cases} 0 \text{ if } u \in C_3^{(1)} \\ 1 \text{ if } u \in C_3^{(2)} \end{cases}$$

where we denote by $C_3^{(1)}$ and $C_3^{(2)}$ the two connected components of G, i.e., the two copies of C_3 in G. That is, h is the indicator function on $C_3^{(1)}$.

We now define f via

$$f : \mathcal{G} \longrightarrow \bigcup_{H \in \mathcal{G}} \mathrm{Fun}(V_H, \mathbb{B}), \qquad f(H) = \begin{cases} h \text{ if } H = G \\ 0 \text{ if } H \neq G \end{cases}$$

where, for a given graph $H = (V_H, E_H)$, we define the function $0 : V_H \to \mathbb{B}$ via $0(u) = 0$ for all $u \in V_H$.

Having constructed our candidate f, we proceed to showing that $\mathcal{S}$ cannot model f. To do so, we first establish the following:

- Denote by $H_1 \dot\cup H_2$ the disjoint union of two graphs. Then $\tilde{\mathcal{S}}(H_1 \dot\cup H_2) = \tilde{\mathcal{S}}(H_1) \dot\cup \tilde{\mathcal{S}}(H_2)$. Here, given two arbitrary functions $f_1 : X \to Z$ and $f_2 : Y \to Z$, we denote by $f_1 \dot\cup f_2$ the natural function $f_1 \dot\cup f_2 : X \dot\cup Y \to Z$.

Depending on your background, this claim might seem trivial. In any case, its proof can be formalized via induction. To do so, denote $H = H_1 \dot\cup H_2$ and denote by $\mathbf{h}_{H',u}^{(t)} \in \mathbb{R}^{d^{(t)}}$ the representation of the vertex $u \in V_{H'}$ of a given graph H' at layer t. We aim to show that

$$\mathbf{h}_{H,u}^{(t)} = \begin{cases} \mathbf{h}_{H_1,u}^{(t)} \text{ if } u \in V_{H_1} \\ \mathbf{h}_{H_2,u}^{(t)} \text{ if } u \in V_{H_2} \end{cases} \tag{1}$$

for all $0 \leq t \leq T$, since this shows that $\tilde{\mathcal{S}}(H)(u) = \mathbf{h}_{H,u}^{(T)} = \mathbf{h}_{H_1,u}^{(T)} = \tilde{\mathcal{S}}(H_1)(u)$ for $u \in V_{H_1}$ and similarly for $u \in V_{H_2}$ (and $V_H = V_{H_1} \dot\cup V_{H_2}$). Equation 1 trivially holds for $t = 0$, since, by assumption, $\mathbf{h}_{H,u}^{(0)} = \mathbf{x}_u = \mathbf{1}^d$ for all graphs and all nodes u. This settles the base case. We proceed with the induction step. Let $1 \leq t \leq T$ and assume that

$$\mathbf{h}_{H,u}^{(t-1)} = \begin{cases} \mathbf{h}_{H_1,u}^{(t-1)} \text{ if } u \in V_{H_1} \\ \mathbf{h}_{H_2,u}^{(t-1)} \text{ if } u \in V_{H_2} \end{cases}$$

Let $u \in V_{H_1}$. We aim to show that $\mathbf{h}_{H,u}^{(t)} = \mathbf{h}_{H_1,u}^{(t)}$. Indeed,

$$\mathbf{h}_{H,u}^{(t)} = \mathsf{MLP}^{(}t) \left(\mathbf{W}_s^{(t)} \mathbf{h}_{H,u}^{(t-1)} + \mathbf{W}_n^{(t)} \sum_{v \in \mathcal{N}_H(u)} \mathbf{h}_{H,v}^{(t-1)} + \mathbf{b}^{(t)} \right)$$

$$\overset{(*)}{=} \mathsf{MLP}^{(}t) \left(\mathbf{W}_s^{(t)} \mathbf{h}_{H_1,u}^{(t-1)} + \mathbf{W}_n^{(t)} \sum_{v \in \mathcal{N}_{H_1}(u)} \mathbf{h}_{H_1,v}^{(t-1)} + \mathbf{b}^{(t)} \right) = \mathbf{h}_{H_1,u}^{(t)}$$

where, for a given graph H', we denote by $\mathcal{N}_{H'}(u)$ the set of neighbors of the node $u \in V_{H'}$ in the graph H'. Here, equality (1) holds due to the induction hypothesis and since, by assumption, $H = H_1 \dot{\cup} H_2$ so that $\mathcal{N}_H(u) = \mathcal{N}_{H_1}(u)$ by assumption. We may conclude claim (1) as stated above.

It only remains to show that $\mathcal{S}$ cannot model f. Indeed, consider any parametrization of $\mathcal{S}$. As we have just shown, $\tilde{\mathcal{S}}(G) = \tilde{\mathcal{S}}(C_3) \dot{\cup} \tilde{\mathcal{S}}(C_3)$. But this means that there exist $u, v \in V_G = C_3^{(1)} \dot{\cup} C_3^{(2)}$ so that u and v are not both in the same connected component and so that $\tilde{\mathcal{S}}(G)(u) = \tilde{\mathcal{S}}(G)(v)$. But, per definition of f as the indicator function on one of the connected components of G, we have $f(u) \neq f(v)$, which establishes the desired result.

(b) We disprove the provided statement by constructing a counter-example. Indeed, choose $T = 1$ as well as $d = d^{(0)} = 1$ and $d^{(1)} = 1$, and consider the parametrization of $\mathcal{S}$ provided by the following matrices and multilayer perceptron.

$$\mathbb{W}_s^{(1)} = [0], \qquad \mathbb{W}_n^{(1)} = [1], \qquad {}^{(1)} = [0],$$

$$\mathrm{MLP}^{(1)} : \mathbb{R} \to \mathbb{R}, \quad \mathrm{MLP}^{(1)}(x) = \sigma(x), \quad \sigma(x) = \begin{cases} 1 \text{ if } x \geq 2 \\ 0 \text{ if } x < 2 \end{cases}.$$

That is, our multilayer perceptron has one neuron and one layer, with a non-linear activation function as prescribed. With this parametrization, we obtain a function defined on the set $\mathcal{G}$ of all bounded-degree graphs;

$$\tilde{\mathcal{S}} : \mathcal{G} \longrightarrow \bigcup_{n=1}^{\infty} \mathbb{B}^n$$

Indeed,

$$\tilde{\mathcal{S}}(G)(u) = \begin{cases} 0 \text{ if } \deg(u) < 3 \\ 1 \text{ if } \deg(u) \geq 3 \end{cases}$$

Applying the provided pooling function, we obtain a function $\tilde{\mathcal{S}}' : \mathcal{G} \to \mathbb{B}$. We now set $f = \tilde{\mathcal{S}}$. Notably, if we label all graphs with $\mathbf{x}_u \equiv 1$, then f is the indicator function of the set of graphs whose maximum degree is at least 3. With this, we have obtained a relevant function $f : \mathcal{G} \to \mathbb{B}$ and parametrized $\mathcal{S}$ so that $\tilde{\mathcal{S}}'$ agrees with f on every bounded-degree graph G.

We can now disprove the provided statement by showing that there does not exist a parametrization of $\mathcal{M}$ that satisfies $\tilde{\mathcal{M}}'(G) = f(G)$ for all bounded-degree graphs G.

Indeed, consider the two graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$, where $V_G = \{1, 2, 3, 4\}$ and $E_G = \{(1, 2), (2, 3), (3, 4)\}$, and $V_H = \{1, 2, 3, 4\}$ and $E_H = \{(1, 2), (2, 3), (3, 4), (1, 3)\}$ (note that,

while we have implicitly chosen a direction for the edges of these graphs, we will ignore this data in our analysis), together with the labels $\mathbf{x}_u \equiv 1$. We aim to show that $\mathcal{M}$ cannot distinguish between these two graphs with the provided labels. Indeed, this would complete our (dis)proof, since $f(G) = 0$ and $f(H) = 1$ (since the degree of all vertices of G is 1 or 2, while $\deg_H(3) = 3$).

To show that, for any parametrization of $\mathcal{M}$, we have $\tilde{\mathcal{M}}(G) = \tilde{\mathcal{M}}(H)$, we show that, for all $0 \leq t \leq T_{\mathcal{M}}$, there exists $\lambda^{(t)} \in \mathbb{R}$ so that $\mathbf{h}_{G,u}^{(t)} = \mathbf{h}_{H,u}^{(t)} = \lambda^{(t)}$ for all $u \in \{1, 2, 3, 4\}$, and proceed by induction. The base case is trivial since $\mathbf{h}_{G,u}^{(0)} = \mathbf{h}_{H,u}^{(0)} = 1$ for all u. For the induction step, let $1 \leq t \leq T_{\mathcal{M}}$ and assume that $\mathbf{h}_{G,u}^{(t-1)} = \mathbf{h}_{H,u}^{(t-1)} = \lambda^{(t-1)}$ for all $u \in \{1, 2, 3, 4\}$. We have that

$$\begin{aligned}
\mathbf{h}_{G,u}^{(t)} &= \mathsf{MLP}^{(t)} \left(\mathbf{W}_s^{(t)} \mathbf{h}_{G,u}^{(t-1)} + \mathbf{W}_n^{(t)} \sum_{v \in \mathcal{N}_G(u)} \frac{\mathbf{h}_{G,v}^{(t-1)}}{|\mathcal{N}_G(u)|} + \mathbf{b}^{(t)} \right) \\
&= \mathsf{MLP}^{(t)} \left(\mathbf{W}_s^{(t)} \lambda^{(t-1)} + \mathbf{W}_n^{(t)} \sum_{v \in \mathcal{N}_G(u)} \frac{\lambda^{(t-1)}}{|\mathcal{N}_G(u)|} + \mathbf{b}^{(t)} \right) \\
&= \mathsf{MLP}^{(t)} \left(\mathbf{W}_s^{(t)} \lambda^{(t-1)} + \mathbf{W}_n^{(t)} \lambda^{(t-1)} + \mathbf{b}^{(t)} \right)
\end{aligned}$$

as well as

$$\mathbf{h}_{G,u}^{(t)} = \mathsf{MLP}^{(t)} \left(\mathbf{W}_s^{(t)} \lambda^{(t-1)} + \mathbf{W}_n^{(t)} \lambda^{(t-1)} + \mathbf{b}^{(t)} \right) =: \lambda^{(t)}$$

which establishes the desired result. (Note that the provided argument generalizes to showing that $\mathcal{M}$ cannot distinguish between any two graphs if all node features are identical.)

Hence, no matter the parametrization of $\mathcal{M}$, we have $\tilde{\mathcal{M}}(G) = \tilde{\mathcal{M}}(H)$. Therefore, $\mathcal{M}$ cannot model f, which differs for G and H.

(c) We disprove the provided statement. Consider the following definitions.

Definition 1. Let x be an n-dimensional one hot vector and let N be a finite set of n-dimensional one hot vectors. Then if $x \in N$, we call the pair (x, N) an n-dimensional neighborhood.

Definition 2. Denote by $\mathcal{N}_n$ the set of n-dimensional neighborhoods. That is,

$$\mathcal{N}_n := \{(x, N) \mid (N, x) \text{ is a neighbourhood of dimension } n\}$$

Since we are restricting to single layer networks, we can understand parametrizations of $\mathcal{S}$ and $\mathcal{M}$ as functions that map $\mathcal{N}_d \to \mathbb{R}^{d^{(1)}}$ via

$$\mathcal{S}' : (\mathbf{x}, N) \longmapsto \mathbf{W}_{\mathcal{S},s}^{(1)} x + \mathbf{W}_{\mathcal{S},n}^{(1)} \sum_{y \in N} y + \mathbf{b}_{\mathcal{S}}^{(1)}$$

and

$$\mathcal{M}' : (\mathbf{x}, N) \longmapsto \mathbf{W}_{\mathcal{M},s}^{(1)} x + \mathbf{W}_{\mathcal{M},n}^{(1)} \sum_{y \in N} \frac{y}{|N|} + \mathbf{b}_{\mathcal{M}}^{(1)}$$

respectively (that is, the networks are fully determined by their values on neighborhoods),[1] post-composed with a MLP. Choosing $d = d^{(0)} = 2$ and $d^{(1)} = 1$ (and $T = 1$, as required in the problem statement), consider the parametrization of $\mathcal{M}$ provided by the following matrices and multilayer perceptron.

$$\mathbf{W}^{(1),s}_{\mathcal{M}} = [0], \quad \mathbf{W}^{(1),n}_{\mathcal{M}} = [1\ 0], \quad \mathbf{b}^{(1)}_{\mathcal{M}} = 0, \quad \mathrm{MLP}^{(1)}_{\mathcal{M}} : \mathbb{R} \to \mathbb{R}, \quad \mathsf{MLP}^{(1)}_{\mathcal{M}}(x) = \sigma(x) - \sigma(-x) = x$$

We now consider a sequence $\{S_i\}_{i=1}^{\infty}$ of sets of neighbourhoods, given by $S_i = \{(x_j^{(i)}, N_j^{(i)})\}_{j=1}^{\infty}$, where $x_j^{(i)} = [1\ 0]$, and

$$N_j^{(i)} = \{x_j^{(i)}, \underbrace{[1\ 0], \ldots, [1\ 0]}_{i\cdot j - 1}, \underbrace{[0\ 1], \ldots, [0\ 1]}_{j}\} = \underbrace{\{\underbrace{[1\ 0], \ldots, [1\ 0]}_{i\cdot j}, \underbrace{[0\ 1], \ldots, [0\ 1]}_{j}\}}_{(i+1)\cdot j}$$

We see that

$$\mathcal{M}'(x_j^{(i)}, N_i^{(j)}) = \frac{i \cdot j}{(i+1)\cdot j} = \frac{i}{i+1}$$

That is, we have identified a countable sequence of sets of neighborhoods S_i (which are also countable) that are mapped to the same value, namely $i/i+1$, by $\mathcal{M}'$. Informally speaking, $\mathcal{M}'$ is "not very injective". Our argument now aims to show that every parametrization of $\mathcal{S}$ is necessarily "more injective" than $\mathcal{M}'$. To proceed our analysis, note that

$$\mathcal{S}' : (x_j^{(i)}, N_j^{(i)}) \longmapsto \mathbf{W}^{(1)}_{\mathcal{S},s} x_j^{(i)} + \mathbf{W}^{(1)}_{\mathcal{S},n} \sum_{y \in N_j^{(i)}} y + \mathbf{b}^{(1)}_{\mathcal{M}} = \mathbf{W}^{(1)}_{\mathcal{S},s} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \mathbf{W}^{(1)}_{\mathcal{S},n} \begin{bmatrix} i \cdot j \\ j \end{bmatrix} + \mathbf{b}^{(1)}_{\mathcal{M}}$$

$$= \mathbf{W}^{(1)}_{\mathcal{S},s} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + j \cdot \mathbf{W}^{(1)}_{\mathcal{S},n} \begin{bmatrix} i \\ 1 \end{bmatrix} + \mathbf{b}^{(1)}_{\mathcal{M}} := \lambda_j^{(i)}$$

For a contradiction, assume that $\mathcal{S}$ is parametrized so that $\mathcal{S}' = \mathcal{M}'$. Then we have that

$$\mathcal{S}'\left(x_j^{(i)}, N_j^{(i)}\right) = \frac{i}{i+1} = \mathcal{M}'\left(x_j^{(i)}, N_j^{(i)}\right) = \mathsf{MLP}^{(1)}_{\mathcal{M}}\left(\lambda_j^{(i)}\right)$$

for all $i, j \in \mathbb{N}$. Therefore, for all $i, i', j, j' \in \mathbb{N}$ with $i \neq i'$, we see that $\mathsf{MLP}^{(1)}_{\mathcal{M}}(\lambda_j^{(i)}) \neq \mathsf{MLP}^{(1)}_{\mathcal{M}}(\lambda_{j'}^{(i')})$, and so $\lambda_j^{(i)} \neq \lambda_{j'}^{(i')}$. We also see that $\mathbf{W}^{(1)}_{\mathcal{S},n}[i\ 1]^T = 0$ for at most one $i \in \mathbb{N}$, and that for all other $i \in \mathbb{N}$, the set $\{\lambda_j^{(i)}\}_j \subseteq \mathbb{R}$ is unbounded, by the definition of $\lambda_j^{(i)}$. Without loss of generality, assume that the two sets $\{\lambda_j^{(1)}\}_j$ and $\{\lambda_j^{(2)}\}_j$ are unbounded in $\mathbb{R}$. Then there exist two sequences $\{k_j^1\}_j \subseteq \mathbb{N}$ and $\{k_j^2\}_j \subseteq \mathbb{N}$ of positive integers so that

$$\lambda^{(1)}_{k_j^1} < \lambda^{(2)}_{k_j^2} < \lambda^{(1)}_{k_{j+1}^1}$$

for all $1 \leq j \leq \infty$. By assumption, we also must have that

$$\mathsf{MLP}^{(1)}_{\mathcal{M}}\left(\lambda^{(1)}_{k_j^1}\right) = 1/2 < 2/3 = \mathsf{MLP}^{(1)}_{\mathcal{M}}\left(\lambda^{(2)}_{k_j^2}\right)$$

for all $1 \leq j \leq \infty$. But, since our multi-layer perceptron $\mathsf{MLP}^{(1)}_{\mathcal{M}}$ is assumed to have $\sigma = \mathsf{ReLU}$ as activation function, it can be represented as a piecewise linear function, with finitely many break points, and so this is impossible.

[1] For the purpose of this particular exercise, this viewpoint will be advantageous because "neighborhoods" are notationally more easily dealt with than "graphs" as input for our GNNs.

Question 2

(a) We define a variation of the 1-dimensional Weisfeiler Leman algorithm, producing codes $\kappa^{(t)}(G)(u)$ for a graph G (with an initial node coloring $c(G) : V_G \to \mathbb{I}^d$) and $t \in \mathbb{N}$, iteratively via the following equation;

$$\kappa^{(t)}(G)(u) = \mathsf{HASH}\left(\kappa^{(t-1)}(G)(u), \left\{\left\{\kappa^{(t-1)}(G)(v) \mid v \in \mathcal{N}_R(u)\right\}\right\}, \left\{\left\{\kappa^{(t-1)}(G)(v) \mid v \in \mathcal{N}_P(u)\right\}\right\}\right)$$

where HASH is an injective function. We set $\kappa^{(0)}(G)(u) = c(G)(u)$.

(b) We establish the provided statement with a proof by induction. Let $\mathcal{A}'$ be a model of $\mathcal{A}$ with T layers and let G be a graph with initial node features $\{x_u = c(G)(c)|u \in V_g\}$. Since by, assumption, $\mathbf{h}_u^{(t)} = c(G)(u) = \kappa(G)(u)$, we clearly have that

$$\mathbf{h}_u^{(0)} \neq \mathbf{h}_v^{(0)} \implies \kappa^{(0)}(G)(u) \neq \kappa^{(0)}(G)(v)$$

for all $u, v \in V_G$, which establishes the base case. For the induction step, let $1 \leq t \leq T$, and assume the induction hypothesis, namely that

$$\mathbf{h}_u^{(t-1)} \neq \mathbf{h}_v^{(t-1)} \implies \kappa^{(t-1)}(G)(u) \neq \kappa^{(t-1)}(G)(v)$$

for all $u, v \in V_G$. We aim to show that

$$\mathbf{h}_u^{(t)} \neq \mathbf{h}_v^{(t)} \implies \kappa^{(t)}(G)(u) \neq \kappa^{(t)}(G)(v)$$

for all $u, v \in V_G$. For a contradiction, assume there exist $u, v \in V_G$ so that $\kappa^{(t)}(G)(u) = \kappa^{(t)}(G)(v)$ and $\mathbf{h}_u^{(t)} \neq \mathbf{h}_v^{(t)}$. Since $\mathbf{h}_u^{(t)} \neq \mathbf{h}_v^{(t)}$, and since the value of $\mathbf{h}^{(t)}(w)$ of a node $w \in V_G$ is completely determined by the triple

$$\left(\mathbf{h}_w^{(t-1)}, \sum_{w' \in \mathcal{N}_R(w)} \mathbf{h}_{w'}^{(t-1)}, \sum_{w' \in \mathcal{N}_P(w)} \mathbf{h}_{w'}^{(t-1)}\right)$$

we must have that

$$\left(\mathbf{h}_u^{(t-1)}, \sum_{u' \in \mathcal{N}_R(u)} \mathbf{h}_{u'}^{(t-1)}, \sum_{u' \in \mathcal{N}_P(u)} \mathbf{h}_{u'}^{(t-1)}\right) \neq \left(\mathbf{h}_v^{(t-1)}, \sum_{v' \in \mathcal{N}_R(v)} \mathbf{h}_{v'}^{(t-1)}, \sum_{v' \in \mathcal{N}_P(v)} \mathbf{h}_{v'}^{(t-1)}\right)$$

as well as

$$\mathsf{HASH}\left(\kappa^{(t-1)}(G)(u), \left\{\left\{\kappa^{(t-1)}(G)(u') \mid u' \in \mathcal{N}_R(u)\right\}\right\}, \left\{\left\{\kappa^{(t-1)}(G)(u') \mid u' \in \mathcal{N}_P(u)\right\}\right\}\right)$$

$$= \mathsf{HASH}\left(\kappa^{(t-1)}(G)(v), \left\{\left\{\kappa^{(t-1)}(G)(v') \mid v' \in \mathcal{N}_R(v)\right\}\right\}, \left\{\left\{\kappa^{(t-1)}(G)(v') \mid v' \in \mathcal{N}_P(v)\right\}\right\}\right)$$

which implies

$$\left(\kappa^{(t-1)}(G)(u), \left\{\left\{\kappa^{(t-1)}(G)(u') \mid u' \in \mathcal{N}_R(u)\right\}\right\}, \left\{\left\{\kappa^{(t-1)}(G)(u') \mid u' \in \mathcal{N}_P(u)\right\}\right\}\right)$$

$$= \left(\kappa^{(t-1)}(G)(v), \left\{\left\{\kappa^{(t-1)}(G)(v') \mid v' \in \mathcal{N}_R(v)\right\}\right\}, \left\{\left\{\kappa^{(t-1)}(G)(v') \mid v' \in \mathcal{N}_P(v)\right\}\right\}\right)$$

since HASH is injective. However, by the induction hypothesis, this is a contradiction.

(c) Let G be a graph with initial node features $\{x_u = c(G)(u) \mid u \in V_G\}$, let $u, v \in V_G$ and let $T \in \mathbb{Z}^+$. We aim to parametrize $\mathcal{A}$ so that, for all $0 \leq t \leq T$,

$$\mathbf{h}_u^{(t)} \neq \mathbf{h}_v^{(t)} \iff \kappa^{(t)}(G)(u) \neq \kappa^{(t)}(G)(v)$$

We have already established that the direction $\implies$ holds for all parametrizations, and now construct a parametrization under which the second direction, $\impliedby$, holds (and that, for convenience, also satisfies $\mathbf{h}_w^{(t)} \in \mathbb{I}^{d^{(t)}}$ for all $w \in V_G$ and $0 \leq t \leq \mathbb{Z}^+$). We do so via strong induction. For the base case, note that

$$\mathbf{h}_u^{(0)} \neq \mathbf{h}_v^{(0)} \iff \kappa^{(0)}(G)(u) \neq \kappa^{(0)}(G)(v)$$

regardless of parametrization, since $\mathbf{h}_w^{(t)} = c(G)(w) = \kappa(G)(w)$ and $c(G)(w) \in \mathbb{I}^{d^{(0)}}$ for all $w \in V_G$.

For the induction step, let $1 \leq t \leq T$ and assume that there exist matrices $\mathbf{W}_S^{(1)}, \ldots, \mathbf{W}_S^{(t-1)}$, $\mathbf{W}_R^{(1)}, \ldots, \mathbf{W}_R^{(t-1)}$, $\mathbf{W}_P^{(1)}, \ldots, \mathbf{W}_P^{(t-1)}$, vectors $\mathbf{b}^{(1)}, \ldots, \mathbf{b}^{(t-1)}$, as well as multi-layer perceptrons $\mathsf{MLP}^{(1)}, \ldots, \mathsf{MLP}^{(t-1)}$ so that $\mathbf{h}_w^{(t-1)} \in \mathbb{I}^{d^{(t-1)}}$ for all $w \in V_G$ and

$$\mathbf{h}_{w_1}^{(t-1)} \neq \mathbf{h}_{w_2}^{(t-1)} \iff \kappa^{(t-1)}(G)(w_1) \neq \kappa^{(t-1)}(G)(w_2)$$

for all $w_1, w_2 \in V_G$.

We aim to find matrices $\mathbf{W}_S^{(t)}$, $\mathbf{W}_R^{(t)}$, $\mathbf{W}_P^{(t)}$ and a vector $\mathbf{b}^{(t)}$, as well as a multi-layer perceptron $\mathsf{MLP}^{(t)}$, so that

$$\mathcal{A}'^{(t)} : \left\{ \left(\mathbf{h}_w^{(t-1)}, \left\{ \left\{ \mathbf{h}_{w'}^{(t-1)} \mid w' \in \mathcal{N}_R(w) \right\} \right\}, \left\{ \left\{ \mathbf{h}_{w'}^{(t-1)} \mid w' \in \mathcal{N}_P(w) \right\} \right\} \right) \;\middle|\; w \in V_G \right\} \longrightarrow \mathbb{I}^{d^{(t)}},$$

$$\left(\mathbf{h}_w^{(t-1)}, \left\{ \left\{ \mathbf{h}_{w'}^{(t-1)} \mid w' \in \mathcal{N}_R(w) \right\} \right\}, \left\{ \left\{ \mathbf{h}_{w'}^{(t-1)} \mid w' \in \mathcal{N}_P(w) \right\} \right\} \right)$$
$$\longmapsto \mathsf{MLP}^{(t)} \left(\mathbf{W}_S^{(t)} \mathbf{h}_w^{(t-1)} + \mathbf{W}_R^{(t)} \sum_{w' \in \mathcal{N}_R(w)} \mathbf{h}_{w'}^{(t-1)} + \mathbf{W}_P^{(t)} \sum_{w' \in \mathcal{N}_P(w)} \mathbf{h}_{w'}^{(t-1)} + \mathbf{b}^{(t)} \right)$$

is injective. Indeed, per the definition of $\kappa^{(t)}$, and by the induction hypothesis, such a choice of matrices would guarantee that

$$\mathbf{h}_{w_1}^{(t)} \neq \mathbf{h}_{w_2}^{(t)} \iff \kappa^{(t)}(G)(w_1) \neq \kappa^{(t)}(G)(w_2)$$

for all $w_1, w_2 \in V_G$.

First, note that the function

$$h_Q : \left\{ \left\{ \mathbf{h}_{w'}^{(t-1)} \mid w' \in \mathcal{N}_Q(w) \right\} \right\} \longmapsto \sum_{w' \in \mathcal{N}_Q(w)} \mathbf{h}_{w'}^{(t-1)}$$

is injective for $Q \in \{R, P\}$, since $\mathbf{h}_{w'}^{(t-1)} \in \mathbb{I}^{d^{(t-1)}}$ are one hot vectors. For $Q \in \{R, P\}$, we now aim to construct a matrix $\mathbf{W'}_Q{}^{(t)}$ and a multi layer perceptron MLP_Q so that

$$f_Q : \sum_{w' \in \mathcal{N}_Q(w)} \mathbf{h}_{w'}^{(t-1)} \longmapsto \mathsf{MLP}_Q \left(\mathbf{W'}_Q{}^{(t)} \cdot \left(\sum_{w' \in \mathcal{N}_Q(w)} \mathbf{h}_{w'}^{(t-1)} \right) \right) \in \mathbb{I}^{m_Q}$$

is injective and maps into $\mathbb{I}^{m_Q}$ where $m_Q \in \mathbb{Z}^+$. We accomplish this in the proceeding paragraphs. Intuitively, our idea is to find a matrix $\mathbf{W'}_Q^{(t)}$ which maps the set of possible input values (the set denoted by r below, which is determined by (G, c) and $\mathcal{A}'^{(t-1)}$) injectively into some Euclidean space. We then use MLP_Q to ensure that the output of f_Q lies in $\mathbb{I}^{m_Q}$.

Denote by M_Q a matrix whose set of rows is equal to the set

$$r = \left\{ \sum_{w' \in \mathcal{N}_Q(w)} \mathbf{h}_{w'}^{(t-1)} \;\middle|\; w \in V_G \right\} \subseteq \mathbb{Z}^{d^{(t-1)}}$$

(Note that this matrix is unique up to permutations of rows.) Consider the vector

$$z = \left[1, n+1, (n+1)^2, \ldots, (n+1)^{d^{(t-1)}}\right]^T$$

where $n = |G|$, so that all entries of M_Q satisfy $0 \leq (M_Q)_{ij} \leq n$. Then, clearly, all components of the vector $b = z \cdot M_Q$ are non-negative and pairwise distinct. Assume, without loss of generality, that $b = (b_1, \ldots, b_{m_Q})$ satisfies $b_1 > b_2 > \cdots > b_{m_Q}$, where we denote by $m_Q = |r|$ the number of rows of M_Q. We can now choose numbers $x_1, \ldots, x_{m_Q} \in \mathbb{R}$ so that

$$\begin{cases} x_i b_j > 1 \text{ if } i \leq j \\ x_i b_j < 1 \text{ if } i > j \end{cases}$$

is satisfied for all $1 \leq i, j \leq m_Q$. In other words, we have chosen $x_1, \ldots, x_{m_Q}$ so that the entries of the matrix $x \cdot b \in \mathbb{R}^{m_Q \times m_Q}$ are greater than 1 on and below the diagonal, and its entries are less than 1 elsewhere. Setting $\mathbf{W'}_Q^{(t)} = x \cdot z$, we have that $\mathbf{W'}_Q^{(t)} \cdot M_Q = \mathbf{W'}_Q^{(t)} \cdot z \cdot M_Q = \mathbf{W'}_Q^{(t)} \cdot b$. With this, we have already identified a matrix that maps the set of possible input values for f_Q, namely the set M_Q, injectively into Eucledian space (specifically, into $\{0,1\}^{m_Q}$).

We now parametrize MLP_Q in such a ways as to ensure that the output of f_Q is a one hot vector. Indeed, recall from our answer to **1(c)** that we can construct a multi-layer perceptron $\mathsf{MLP}_{\chi,d,b}$ that approximates the indicator function on $[1, \infty)$ arbitrarily well. To utilize this result, let $\delta_Q \in \mathbb{R}^+$ be so that for all entries e_i^j of $X_Q := \mathbf{W'}_Q^{(t)} \cdot M_Q$, if $e_i^j > 1$, then $e_i^j > 1 + \delta_Q$. Then we have that

$$\mathsf{MLP}_{\chi,\delta_Q,1}(x) = \begin{cases} 0 & \text{if } x \leq 1 \\ \dfrac{x-1}{\delta_Q} & \text{if } 1 < x < 1 + \delta_Q \\ 1 & \text{if } x \geq 1 + \delta_Q \end{cases}$$

and therefore, applying $\mathsf{MLP}_{\chi,\delta_Q,1}$ entry-wise to X_Q, obtain a lower triangular matrix $\mathsf{MLP}_{\chi,\delta_Q,1}(X_Q) \in \{0,1\}^{m_Q \times m_Q}$. We now construct MLP_Q by composing $\mathsf{MLP}_{\chi,\delta_Q,1}$ with another multi layer perceptron to turn our matrix into a non-singular matrix of one hot rows. Indeed, let MLP_Q be the multi layer perceptron represented by

$$\mathsf{MLP}_Q : \mathbb{R}^{d_Q} \longrightarrow \mathbb{R}^{d_Q}, \qquad \begin{bmatrix} x_1 \\ \vdots \\ x_{m_Q} \end{bmatrix} \longmapsto \mathsf{MLP}_Q^* \left(\begin{bmatrix} \mathsf{MLP}_{\chi,\delta,1}(x_1) \\ \vdots \\ \mathsf{MLP}_{\chi,\delta,1}(x_{m_Q}) \end{bmatrix} \right)$$

where MLP_Q^* denotes the single-layer multi-layer perceptron of width d^{d_Q} represented by

$$\mathsf{MLP}_Q^* : \mathbb{R}^{d_Q} \longrightarrow \mathbb{R}^{d_Q}, \qquad \begin{bmatrix} x_1 \\ \vdots \\ x_{m_Q} \end{bmatrix} \longmapsto \begin{bmatrix} \sigma(x_1) \\ \sigma(x_2 - x_1) \\ \sigma(x_3 - x_2) \\ \vdots \\ \sigma(x_{m_Q} - x_{m_Q - 1}) \end{bmatrix}$$

Clearly, we have that $\mathsf{MLP}_Q(X_Q) = \mathrm{id}_{m_Q}$. But this implies that f_Q is injective, and that $f(M_Q) \subseteq \mathbb{I}^{m_Q}$.

With this, we have now obtained two matrices, ${\mathbf{W}'_R}^{(t)}$ and ${\mathbf{W}'_P}^{(t)}$, that allow us to construct the required parametrizing matrices, bios vector and multi-layer perceptron for A. Indeed, we set $\mathbf{W}_S^{(t)}$, $\mathbf{W}_R^{(t)}$, and ${\mathbf{W}'_P}^{(t)}$ to be $d^{(t-1)} \times (d^{(t-1)} + m_R + m_P)$-matrices, where the first $d^{(t-1)}$ rows of $\mathbf{W}_S^{(t)}$ form the identity matrix, the rows of $\mathbf{W}_R^{(t)}$ from $d^{(t-1)} + 1$ to $d^{(t-1)} + m_R$ form the matrix ${\mathbf{W}'_R}^{(t)}$, the rows of $\mathbf{W}_P^{(t)}$ from $d^{(t-1)} + m_R + 1$ to $d^{(t-1)} + m_R + m_P$ form the matrix ${\mathbf{W}'_P}^{(t)}$, and everything else is filled with zeroes. Further, we set $\mathbf{b}^{(t)}$ to be a $(d^{(t-1)} + m_R + m_P)$-dimensional vector of just zeroes, and set MLP^t to be a combination of the perceptrons MLP_R and MLP_P and $\mathsf{MLP}_*^{(t)}$, as defined below.

$$\mathsf{MLP}^t : \mathbb{R}^{d^{(t-1)} + m_R + m_P} \longrightarrow \mathbb{R}^{d^{(t-1)} \cdot m_R \cdot m_P},$$

$$\begin{bmatrix} x_1 \\ \vdots \\ x_{d^{(t-1)} + m_R + m_P} \end{bmatrix} \longmapsto \mathsf{MLP}_*^{(t)} \left(\begin{bmatrix} x_1 \\ \vdots \\ x_{d^{(t-1)}} \\ \mathsf{MLP}_R \left(\begin{bmatrix} x_{d^{(t-1)}+1} \\ \vdots \\ x_{d^{(t-1)}+m_R} \end{bmatrix} \right) \\ \mathsf{MLP}_P \left(\begin{bmatrix} x_{d^{(t-1)}+m_R+1} \\ \vdots \\ x_{d^{(t-1)}+m_R+m_P} \end{bmatrix} \right) \end{bmatrix} \right)$$

With this, the input to $\mathsf{MLP}_*^{(t)}$ (that occurs during computation) is guaranteed to always be an element of $\mathbb{I}^{d^{(t-1)}} \times \mathbb{I}^{m_R} \times \mathbb{I}^{m_P}$, i.e., three one hot vectors. We finally set $\mathsf{MLP}_*^{(t)}$ to be an injection mapping $\mathbb{I}^{d^{(t-1)}} \times \mathbb{I}^{m_R} \times \mathbb{I}^{m_P}$ into $\mathbb{I}^{d^{(t-1)} \cdot m_R \cdot m_P}$. This can be achieved with a single-layer perceptron of width $d^{(t-1)} \cdot m_R \cdot m_P$, where all neurons are of the form $\sigma(x_i + x_j + x_k - 2)$ for $1 \leq i \leq d^{(t-1)}$, $d^{(t-1)} + 1 \leq j \leq d^{(t-1)} + m_R$ and $d^{(t-1)} + m_R + 1 \leq k \leq d^{(t-1)} + m_R + m_P$.

With this setup, $\mathcal{A}'^{(t)}$ maps

$$\left(\mathbf{h}_w^{(t-1)}, \left\{ \left\{ \mathbf{h}_{w'}^{(t-1)} \mid w' \in \mathcal{N}_R(w) \right\} \right\}, \left\{ \left\{ \mathbf{h}_{w'}^{(t-1)} \mid w' \in \mathcal{N}_P(w) \right\} \right\} \right) \longmapsto$$

$$\mathsf{MLP}_*^{(t)} \begin{bmatrix} \mathbf{h}_w^{(t-1)} \\ f_R \left(\sum_{w' \in \mathcal{N}_R(w)} \mathbf{h}_{w'}^{(t-1)} \right) \\ f_P \left(\sum_{w' \in \mathcal{N}_P(w)} \mathbf{h}_{w'}^{(t-1)} \right) \end{bmatrix} = \mathsf{MLP}_*^{(t)} \begin{bmatrix} \mathbf{h}_w^{(t-1)} \\ f_R \left(h_R \left(\left\{ \left\{ \mathbf{h}_{w'}^{(t-1)} \mid w' \in \mathcal{N}_R(w) \right\} \right\} \right) \right) \\ f_P \left(h_P \left(\left\{ \left\{ \mathbf{h}_{w'}^{(t-1)} \mid w' \in \mathcal{N}_P(w) \right\} \right\} \right) \right) \end{bmatrix} \in \mathbb{I}^{d^{(t-1)} \cdot m_R \cdot r_P}$$

which is injective, since $\mathsf{MLP}_*^{(t)}$, f_R, f_P, h_R and h_P are injective, and maps into $\mathbb{I}^{d^{(t)}}$, where $d^{(t)} = d^{t-1} \cdot m_R \cdot r_P$, by our construction. This concludes the induction step.

(d) Due to Theorem 4.2 in the paper (Barcelo et al., 2020), we know that each unary FOC_2 formula can be captured by a parametrization of model $\mathcal{B}$. We now adopt a second result from this paper, Theorem 4.1, to model $\mathcal{A}$. Indeed, consider the following definition of a logic L.

Definition. A formula in L is constructed as follows:

1. if r is one of the base colors (that is, there exists $u \in V_G$ with $r = c(G)(u)$), then r is a formula in L, and
2. if C and D are formulas, E_R and E_P are the two relations

$$E_R(u,v) \iff v \in \mathcal{N}_R(u), \qquad E_P(u,v) \iff v \in \mathcal{N}_P(u),$$

respectively, and n is a natural number, then $C \sqcap D$, $C \sqcup D$, $\exists^{\geq n} E_R$ and $\exists^{\geq n} E_P$ are formulas in L.

We define when a node v in a graph G *satisfies* a formula α in L, denoted by $v \models \alpha$, recursively as follows:

1. if $\alpha = r$, then $v \models \alpha$ if and only if r is the base color of v in G, that is, $c(G)(v) = r$,
2. if $\alpha = C \sqcap D$, then $v \models \alpha$ if and only if $v \models C$ and $v \models D$ (and similarly with $C \sqcup D$ and $\neg C$), and
3. if $\alpha = \exists^{\geq n} E_R.C$, then $v \models \alpha$ if and only if the set of nodes $\{u | u \in \mathcal{N}_R(v) \text{ and } v \models C\}$ has at least size n, as well as if $\alpha = \exists^{\geq n} E_P.C$, then $v \models \alpha$ if and only if the set of nodes $\{u | u \in \mathcal{N}_P(v) \text{ and } v \models C\}$ has at least size n.

We now adapt the proof of Theorem 4.1 of (Barcelo et al., 2020) to this logic, proving that the model archiecture $\mathcal{A}$ with T layers of the form

$$\mathbf{h}_u^{(t)} = \mathsf{MLP}^{(t)} \left(\mathbf{W}_s^{(t)} \mathbf{h}_u^{(t-1)} x + \mathbf{W}_{\mathcal{Q}}^{(t)} \sum_{v \in \mathcal{Q}(u)} \mathbf{h}_v^{(t-1)} + \mathbf{W}_{\mathcal{V}}^{(t)} \sum_{v \in \mathcal{V}(u)} \mathbf{h}_v^{(t-1)} + \mathbf{b}^{(t)} \right)$$

can capture an arbitrary formula α from the logic L, by choosing suitable depth T and matrices $\mathbf{W}_{\int}^{(t)}$, $\mathbf{W}_{\mathcal{Q}}^{(t)}$, $\mathbf{W}_{\mathcal{V}}^{(t)}$, and the bios vector $\mathbf{b}^{(t)}$. Indeed, the proof provided in (Barcelo et al., 2020), which can be found in appendix B, starting on the lower half of page 6, can be adapted almost line by line. The proof is constructive and provides a homogeneous parametrization of the model; that is, all layers use the same parameters. The proof first enumerates the subformulas of α by $\mathsf{sub}(\alpha) = (\alpha_1, \ldots, \alpha_K)$. All feature vectors are elements of $\mathbb{R}^K$. The idea is that $(\mathbf{h}_u^{(t)})_i = 1$ only if the subformula α_i holds true for the node u.

We adapt the notation used in the proof, denoting $\mathbf{C} = \mathbf{W}_{\int}^{(t)}$, $\mathbf{A}_1 = \mathbf{W}_{\mathcal{Q}}^{(t)}$, $\mathbf{A}_2 = \mathbf{W}_{\mathcal{V}}^{(t)}$, and $\mathbf{b} = \mathbf{b}^{(t)}$. Then, we construct the matrices $\mathbf{C}$, $\mathbf{A}_1$, $\mathbf{A}_2$ and $\mathbf{b}$ as described in the cases 1 through 5 in the proof. We only need to adapt case 5, which we split into two:

Case 5a: if $\alpha_i = \exists^{\geq n} E_R.\alpha_j$, then $(\mathbf{A}_1)_{ij} = 1$ and $b_i = -n + 1$,
Case 5b: if $\alpha_i = \exists^{\geq n} E_P.\alpha_j$, then $(\mathbf{A}_2)_{ij} = 1$ and $\mathbf{b}_i = -n + 1$.

Further, as in the original proof, all other unset values of $\mathbf{C}$, $\mathbf{A}_1$, $\mathbf{A}_2$ and $\mathbf{b}$ are set to 0. Lastly, also as in the original proof, we set $\mathsf{MLP}^{(t)}$ to be the *hard-sigmoid*, defined by $\mathsf{MLP}^{(t)}(u) = \min(\max(0, u), 1)$.

It is now easy to see the the rest of the proof of Theorem 4.1 applies one to one to our setup. A slight modification is necessary only for the very last part, at the top of page 8. However, showing that if $\alpha_i = \exists^{\geq n} E_R$, or $\alpha_i = \exists^{\geq n} E_P$, then $(\mathbf{h}_u^{(T)})_i = 1$ if and only if $(G, u) \vDash \exists^{\geq n} E_R$, or $(G, u) \vDash \exists^{\geq n} E_P$, respectively, still relies on exactly the same argument, except that it is now split into two cases, as was case 5 above.

For reference, the paper (Abboud et al., 2023) similarly adapts the proof of Theorem 4.1 with a slight modification. The relevant section is appendix C, especially the lower half of page 19.

With this we have established two precise results contrasting the expressive power of the model architectures A and B.

Question 3

While message-passing graph neural networks (MPGNNs) have emerged as the dominant paradigm of graph representation learning, notable limitations in the expressiveness of these models have been established. In particular, researchers have identified over-smoothing and over-squashing as so-called information bottle-necks—limiting the representative power of message-passing architectures. This work proposes a new MPGNN architecture that draws inspiration from graph theory and the work of (Gilmer et al., 2017), designed specifically for learning on molecules, and argues that this architecture addresses these two limitations.

Let $G = (V_G, E_G)$ be a graph with features $\{\mathbf{x}_u\}_u$. Over-smoothing describes the observation that the representations $\mathbf{h}_u^{(t)}$ and $\mathbf{h}_v^{(t)}$ of two nodes become increasingly indistinguishable after several iterations of message passing. Indeed, (Li et al., 2018) showed that if G is connected and not bi-partite, the sequences $\{\mathbf{h}_u^{(t)}\}_{t\in\mathbb{Z}}$ converge to a common value for all $u \in V_G$ (Theorem 1). This has obvious consequences for the choice of model depth, and (Li et al., 2018) demonstrate, utilizing Zachary's karate club dataset, that relatively few convolutional layers can lead to a mixing of node information and a sharp decrease in classification accuracy.

The issue of choosing model depth is also deeply connected to over-squashing. This phenomen stems from the fact that the size of the receptive field of a node $u \in V_G$ in a graph — that is, the number of nodes within a distance t of u — grows exponentially with t. However, to allow for a reasonable number of trainable parameters, the length of node vectors $\mathbf{h}_u^{(t)}$ is usually bounded. Consequently, MPGNNs "perform poorly when the prediction task depends on long range interaction" (Alon and Yahav, 2021). Ramifications to over-squashing include adding *supersource nodes* (Scarselli et al., 2008) and adding *virtual edges*, both aiming to capture long range interactions (Gilmer et al., 2017).

Given that graph representation learning has applications in a variety of fields, there is interest in both models that perform excellently on a multitude of kinds of graphs (drawing from chemistry, social networks, code syntax, and more) (Errica et al., 2020, Abboud et al., 2023), and in models that are designed to solve a specific task. The latter is the case with supervised learning on molecules. (Gilmer et al., 2017) argue that research in molecular learning should focus on finding particularly suitable variants of the message passing approach, trained on selected datasets.

Following this approach, we propose a new model architecture that is particularly expressive on molecular graphs and study it in the context of the dataset NCI1 from the TUDataset collection (Morris et al., 2020). The NCI1 dataset contains 4110 graphs with an average of ~ 30 nodes and an average node degree of 2. Its task is graph-level, namely the prediction of anti-lung-cancer activity in biochemical compounds. Before introducing the model architecture, we introduce the following novel, graph-theoretic definitions.

Definition. Let $G = (V_G, E_G)$ be a graph. We say that a set $H \subseteq V_G$ of nodes of G is **biconnected** if it has at least three nodes, is connected, and remains connected even when one node is removed from H.

Definition. Let $G = (V_G, E_G)$ be graph and $u \in V_G$ a node in G. We say that u is an **anker node** of G if $\deg(u) \geq 3$ and at most one of the neighbors of u has degree greater than 1. We then

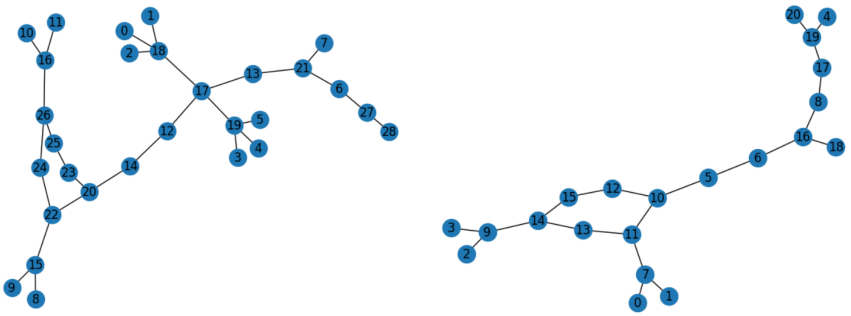

Figure 1: To the left, we see graph G_1. Here, {20, 22, 23, 24, 25, 26} is the single bi-connected component, {16, 26, 10, 11}, {0, 1, 2, 17, 18}, {3, 4, 5, 17, 19}, and {8, 9, 22, 15} form forks, and {21, 7}, {17, 21, 13}, {27, 28, 21, 6} as well as {17, 12, 20, 14} are branches.
To the right, we see graph G_2. Here, {10, 11, 12, 13, 14, 15} is the single bi-connected component, {0, 1, 11, 7} and {9, 2, 3, 14} are the forks, and {4, 5, 6, 8, 10, 16, 17, 18, 19, 20} forms a branch.

call the set $\{u\} \cup \mathcal{N}(u)$ a **fork** of G.

Definition. Let $G = (V_G, E_G)$ be a graph and let $S \subseteq V_G$ be a non-empty set of notes of G. Further, denote by $S' \subseteq S$ the set of nodes in S with degree at most 2. We say that S is a **branch** of G if S' is connected and every node in $S \setminus S'$ is connected to at least one node in S', and, further, S is not contained in a fork or bi-connected component of G.

Definition. Let $G = (V_G, E_G)$ be a graph. We define the **component graph** C_G of G as follows: Each bi-connected component, fork, and branch of G represents a node of C_G (that is, nodes $u \in C_G$ are subsets $u \subseteq V_G$), and two nodes of C_G are connected by an edge if their intersection is non-empty.

These definitions become intuitive by looking at specific examples. Figure 1 shows two graphs from the NCI1 dataset, and lists their bi-connected components, forks and branches. Figure 2 shows the corresponding component graphs. We can see that the component graphs encode key features of the original graphs, capturing the structure of the graphs on a global level. Further inspection of the NCI1 dataset shows that the three types of subgraphs we defined above feature prominently accross the dataset. Indeed, it can be verified empirically that for each connected component H of a graph of the NCI1 dataset, the component graph C_H of H is connected.

We now introduce our model architecture $\mathcal{A}$. Let G be a graph. The first two layers of $\mathcal{A}$ consisting of standard convolutional layers, so that

$$\mathbf{h}_u^{(t)} = \mathsf{MLP}_t(\mathbf{W}_s^{(t)}\mathbf{h}_u^{(t-1)} + \mathbf{W}_n^{(t)} \sum_{v \in \mathcal{N}_G(u)} \mathbf{h}_v^{(t-1)}) \tag{2}$$

for $1 \leq t \leq 2$. Then, for each node $u_C \in C_G$, we calculate the mean $\sum_{u \in u_C} \mathbf{h}_u^{(2)}/|u_C|$, obtaining node features on the component graph C_G of G. We then apply another two standard convolutional layers to the component graph, computing the representations $\mathbf{h}_{u_C}^{(t)}$ for $3 \leq t \leq 4$. MLP_4 should have a scalar value in the interval $[0, 1]$ as output (indicating whether the compound representated by the graph G provides anti-lung-cancer activity). In our implementation, the hidden dimension across the MPGNN is 40 and all multi-layer perceptrons consist of only one layer and use ReLU as

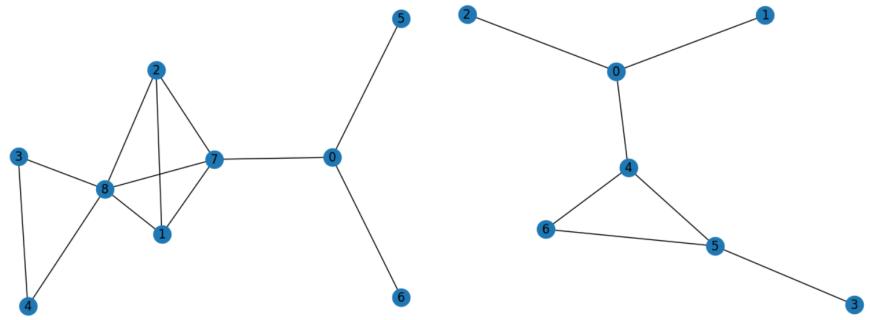

Figure 2: To the left we see the component graph of G_1. The following dictionary describes its labels: {0: {20, 22, 23, 24, 25, 26}, 1: {0, 1, 2, 17, 18}, 2: {3, 4, 5, 17, 19}, 3: {27, 28, 21, 6}, 4: {21, 7}, 5: {8, 9, 22, 15}, 6: {16, 26, 10, 11}, 7: {17, 12, 20, 14}, 8: {17, 21, 13}}.
To the right we see the component graph of G_2. The following dictionary describes its labels: {0: {10, 11, 12, 13, 14, 15}, 1: {0, 1, 11, 7}, 2: {9, 2, 3, 14}, 3: {17, 19, 4, 20}, 4: {16, 10, 5, 6}, 5: {8, 17, 19, 16}, 6: {16, 18}}.

activation function.

Our model architecture $\mathcal{A}$ utilizes the fact that C_G encodes global information of the graph G and has significantly fewer nodes. This aids long range interactions between nodes of G. For example, to transfer information from node 3 to node 4 in graph G_2 (Figure 1), a traditional MPGNN would need depth at least 12. Our model, on the other hand, can transfer information between these two nodes with only four layers (crucially, two layers on the component graph C_{G_2}). More generally, out of 1393 connected graphs in the NCI1 dataset, 497 have a component graph with diameter 2, and the average diameter of component graphs is ~ 3.1. This allows $\mathcal{A}$ to meaningfully bring together information from all nodes of the graph while avoiding over-smoothing and over-squashing, and hence allowing it to represent new classes of functions.

As part of this work, $\mathcal{A}$ has been implemented using pytorch geometric and trained on the NCI1 dataset. Following the evaluation framework outlined by (Errica et al., 2020), an external tenfold split was applied to the dataset. The dataset was preprocessed, computing the component graph defined above for each graph in NCI1. For efficient training, this data had to be referenced by the pytorch implementation of A. Due to time constraints and the technical implementation, this meant that the model could not be trained with batches, which impacted accuracy significantly. A standard MPGNN, with four convolutional layers as described in equation 2, achieved an average accuracy of 41.56, while model A achieved an average accuracy of 46.24.

With this, our model achieves a sigfnificant advantage over the traditional architecture. However, this work should be continued by adapting the code to work with mini batches, and by further tuning hyperparameters such as number of layers and hidden dimension of the MPGNN. The current code base can be found here. Furhter, the expressivity of A may be examined theoretically using some of the tools available in the literature, such as studying the Jacobi of node representations with respect to initial node features, which also helps study over-squashing (Topping et al., 2022). Whether our proposed model A overcomes over-squashing may be further examined by using a synthetic datasets, such as the h-Proximity graphs introduced in (Abboud et al., 2022). Last but not least, it is of interest to compare how A performs on chemical datasets beyond NCI1.